SMC-TR-06-06

AEROSPACE REPORT NO.
TR-2006(8550)-1

Reducing Software Acquisition Risk: Best Practices for the Early Acquisition Phases

31 January 2006

Prepared by

R. J. ADAMS, S. ESLINGER, K. L. OWENS, and M. A. RICH
Software Engineering Subdivision
Computers and Software Division

Prepared for

SPACE AND MISSILE SYSTEMS CENTER
AIR FORCE SPACE COMMAND
330 W. Orbital Loop
Los Angeles Air Force Base, CA 90245

Engineering and Technology Group

THE AEROSPACE CORPORATION
El Segundo, California

This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. FA8802-04-C-0001 with the Space and Missile Systems Center, 2430 E. El Segundo Blvd., Los Angeles Air Force Base, CA 90245. It was reviewed and approved for The Aerospace Corporation by Mary A. Rich, Principal Director, Software Engineering Subdivision, Computers and Software Division. Michael Zambrana was the project officer for the Mission-Oriented Investigation and Experimentation (MOIE) program.

This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

Michael Zambrana
SMC/AXE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (*DD-MM-YYYY*)	2. REPORT TYPE	3. DATES COVERED *(From - To)*
31/01/06		

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Reducing Software Acquisition Risk: Best Practices for the Early Acquisition Phases	5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
R. J. Adams, S. Eslinger, K. L. Owens, and M. A. Rich	5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
The Aerospace Corporation El Segundo, CA 90245-4691	TR-2006(8550)-1
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
Space and Missile Systems Center Air Force Space Command 330 W. Orbital Loop Los Angeles Air Force Base, CA 90245	SMC 11. SPONSOR/MONITOR'S REPORT NUMBER(S) SMC-TR-06-06

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The acquisition of large, complex software-intensive systems has historically been fraught with major problems, including performance deficiencies, extensive software defects, and cost and schedule overruns. Furthermore, these software development problems usually do not manifest themselves until late in the development life cycle. This has led to the question of what can be done early in the acquisition life cycle to reduce the risk of these software problems occurring later in the program. This paper addresses software acquisition best practices for the early National Security Space (NSS) acquisition life cycle phases (Pre-KDP A and Phase A). These best practices are targeted toward reducing downstream software development risk, and thereby improving mission success for software-intensive space systems.

15. SUBJECT TERMS

Software, Software Acquisition, Software Engineering, Best Practices

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON S. Eslinger
a. REPORT UNCLASSIFIED	b. ABSTRACT UNCLASSIFIED	c. THIS PAGE UNCLASSIFIED		38	19b. TELEPHONE NUMBER *(include area code)* (310)336-2906

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

Acknowledgements

This work was funded under the Mission-Oriented Investigation and Experimentation (MOIE) research program (Software Acquisition Task). The authors would like to thank Mike Zambrana (SMC/AXE) for his strong support of their work.

Contents

Figures

1. Introduction

Proposed changes in acquisition policy have solidified over the past several years with the establishment of new versions of DODD 5000.1 [DOD1] and DODI 5000.2 [DOD2], a new capability-based process for requirements generation, and new acquisition policy specific to National Security Space (NSS) systems (NSS Acquisition Policy 03-01) [USECAF1]. Many mandates requiring the use of acquisition reform practices defined during the 1990s have been lifted (e.g., prohibitions against use of military standards, restrictions on quantity and type of technical deliverable documentation, contractor Total System Performance Responsibility).

With the focus on new acquisition practices, the opportunity exists for identifying and implementing a comprehensive set of software acquisition best practices designed to reduce risk in the acquisition of software-intensive systems. In 2004, the authors published a technical report [ADAMS1] that described such a comprehensive set of software acquisition best practices for a system development contract.

The acquisition of large, complex software-intensive systems has historically been fraught with major problems, including performance deficiencies, extensive software defects, and cost and schedule overruns. Furthermore, these software development problems usually do not manifest themselves until late in the development life cycle. This has led to the question of what can be done early in the acquisition life cycle to reduce the risk of these software problems occurring later in the program. This is an especially important question for the United States Air Force (USAF) Space and Missile Systems Center (SMC), since there have recently been several large, complex, software-intensive SMC programs in the early acquisition phases. To answer this question, the authors have identified an additional set of software acquisition best practices for the early acquisition life cycle phases that extend those published in 2004. These new software acquisition best practices are targeted toward reducing downstream software development risk, and thereby improving mission success for software-intensive space systems.

2. Definition of Software Acquisition Best Practices

2.1 Software Acquisition Processes

The term "software acquisition" is defined to mean the set of processes (i.e., the methods, tools, techniques, procedures, etc.) used by the government to acquire the software portion of software-intensive systems. Software acquisition covers the activities of the entire program life cycle, from the identification of needed capability through system retirement, including pre-contract award and post-contract award activities. The term "software engineering," on the other hand, is defined to be the set of processes used by the developers to build the software portion of software-intensive systems. Figure 1 illustrates the different domains of software acquisition and software engineering.

One of the principal components of a successful software development project is the quality of the software engineering processes used. This statement is based on the well-established fact that the quality of a software product is highly dependent upon the quality of the processes used to develop and maintain that product [PAULK, p. 8]. However, software acquisition processes are also very influential in achieving a successful software development project. The software acquisition processes used can positively encourage, or adversely constrain, the developers in their application of high-quality software engineering processes. The application of software acquisition best practices, therefore, can reduce risk in the acquisition of software-intensive systems.

2.2 Characteristics of Software Acquisition Best Practices

Software acquisition best practices are, by definition, practices that people with recognized software acquisition expertise have identified through experience as being significant contributors to the successful acquisition of software-intensive systems. Both negative experiences and positive experiences on past programs can be used to identify software acquisition best practices. However, care must be taken when using negative experiences so as not to be trapped by logical fallacies. If practice "A" is used, and the resulting experience "B" is not desirable, this does not imply that if practice "A" is not used, then "B" will not occur and the resulting experience will be desirable. (A => B is not equivalent to Not A => Not B.)

Since best practices are experientially based, a best practice has never been proven to be "best" in any analytical sense. Best practices also never form an exhaustive set; there is always the possibility of additional best practices being identified. In addition, they are not static. Best practices change based on new experiences and new technologies.

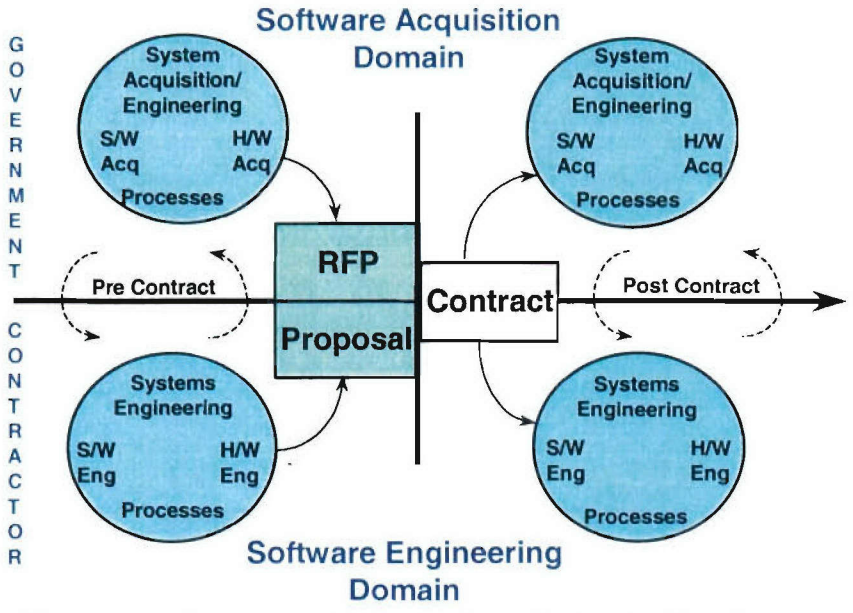

Figure 1. Software Acquisition and Software Engineering Domains.

A comprehensive set of software acquisition best practices must provide a consistent and integrated approach to software acquisition throughout the program life cycle, both pre- and post-contract award. In addition, because software always exists within the context of the system, the software acquisition best practices must be consistent and integrated with a comprehensive set of system acquisition best practices. The set of best practices must also be suitable for acquiring today's complex software systems that will be developed using the latest software development process and product technologies. Finally, the software acquisition best practices must be able to be applied in the context of the specific acquisition environment and be consistent with the applicable system and software acquisition policies and regulations at all levels of that acquisition environment (e.g., DOD, NSS, AF, SMC).

Today's software-intensive space systems are large systems with multiple-satellite constellations and multiple ground elements, frequently worldwide. These systems involve complex combinations of hardware and software with complex external and internal interfaces. They are usually unprecedented and have high reliability and integrity requirements. The size of the software in space systems now under development is on the order of 10^5 Source Lines of Code (SLOC) onboard and 10^6 to 10^7 SLOC on the ground. In addition, large teams of contractors develop these systems. Space systems software acquisition best practices must be able to effectively support the acquisition of these systems.

Based on experiences from supporting USAF SMC and the National Reconnaissance Office (NRO) in the acquisition of software-intensive space systems over a 20-year period, the authors have developed a comprehensive set of space system software acquisition best practices satisfying the above criteria. The authors have over 60 collective years of experience in the acquisition of software-intensive space systems while at The Aerospace Corporation, in addition to substantial prior experience developing space and similar software-intensive systems in industry. While the authors developed these software acquisition best practices based on experiences in the space

system domain, the best practices can be applied to all acquisitions of large, complex software-intensive systems in domains that require high reliability and integrity.

3. The NSS Acquisition Life Cycle

Figure 2 shows the NSS and DOD acquisition life cycles, with corresponding acquisition phases and milestones aligned. The Acquisition Program Baseline (APB) is approved at Key Decision Point (KDP) B for NSS programs (Milestone B for DOD programs). At that point, the program begins design, implementation, and operations, which for NSS programs, spans three acquisition phases (Phase B, Preliminary Design; Phase C, Complete Design; and Phase D, Build and Operations). The early NSS acquisition life cycle phases consist of Pre-KDP A (Concept Studies) and Phase A (Concept Development).

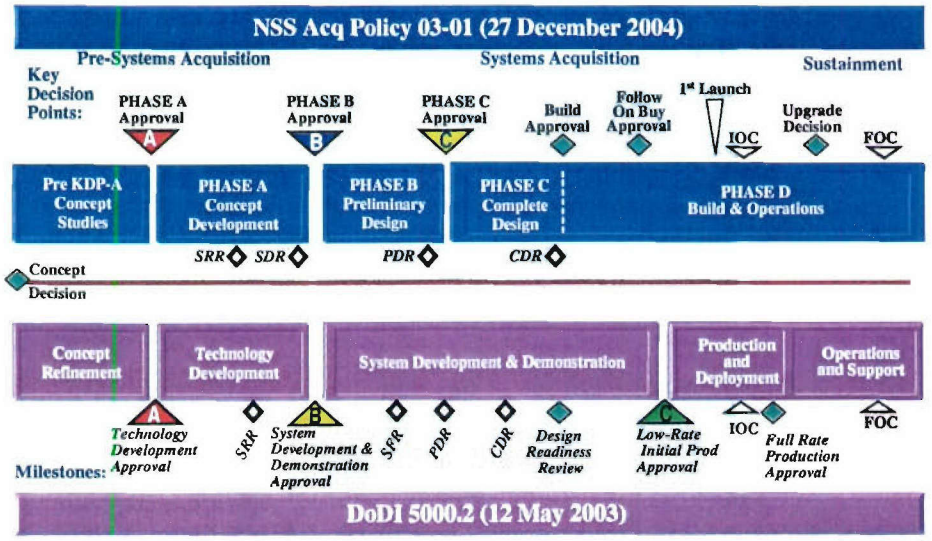

Figure 2. NSS and DOD Acquisition Life Cycles.

The software acquisition best practices published in 2004 [ADAMS1] address best practices for pre- and post-contract award acquisition activities for a large, complex software-intensive space system development contract. Those best practices therefore primarily relate to NSS acquisition phases B, C, and D (or the corresponding DOD acquisition phases shown in Figure 2, for programs adhering to the DOD 5000 series regulations). This paper addresses software acquisition best practices for the early NSS acquisition phases, Pre KDP-A (Concept Studies) and Phase A (Concept Development). These best practices are equally applicable to the corresponding early acquisition phases for DOD 5000 programs.

Most other organizations (such as NASA) that are responsible for large, complex software-intensive system acquisitions have similar acquisition life cycle models whose acquisition phases

can be aligned with the NSS and DOD acquisition phases shown in Figure 2. The software acquisition best practices presented in this paper and in [ADAMS1] can be applied to acquisitions conducted by those other organizations as well.

4. Software Acquisition Best Practices for the Early Acquisition Phases

The software acquisition best practices for Pre-KDP A and Phase A are described in more detail below. This report, however, is not intended to be a tutorial in the application of these best practices. Rather, it briefly describes the best practices, the rationale for their inclusion, and essential elements for their effective application. It should be noted that the recommended software acquisition best practices are not independent of each other; in fact, effective use of a particular software acquisition best practice may require concurrent use of other software acquisition best practices.

4.1 Pre-KDP A Space System Software Acquisition Best Practices

The Pre-KDP A activities focus on sufficiently maturing the program concepts to be ready to enter Phase A, the Concept Development phase. Figure 3 depicts the categories of software acquisition best practices for the activities in this phase. There are six categories of best practices addressing: defining the program life cycle (see paragraph 4.1.1), the initial system definition (see paragraph 4.1.2), the initial government architecture concepts (see paragraph 4.1.3), the initial government cost and schedule baselines (see paragraph 4.1.4), executable program evolutions (see paragraph 4.1.5), and the global acquisition and Test and Evaluation (T&E) strategies (see paragraph 4.1.6). The software acquisition best practices in each of these categories are discussed below.

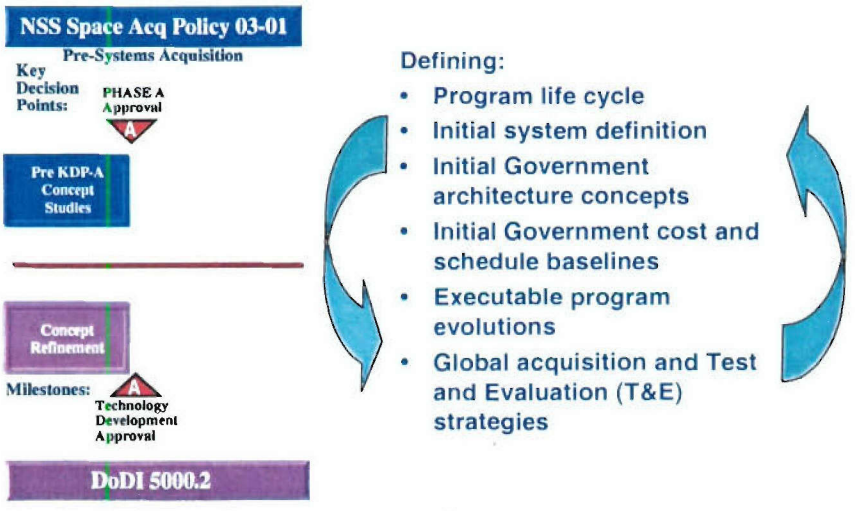

Figure 3. Pre-KDP A Categories of Software Acquisition Best Practices.

4.1.1 Defining the Program Life Cycle

There are three software acquisition best practices for defining the program life cycle: a) using a software-friendly acquisition model, b) choosing software-friendly points in the life cycle for contract actions, and c) tailoring the acquisition model for the software-intensive system.

a) Use a Software-Friendly Acquisition Model

This best practice involves including consideration of the software development effort in the selection of the acquisition life cycle model to be used. There are two basic acquisition life cycle models that can be used for NSS and DOD systems: single step and evolutionary acquisition. History has shown that it is very difficult to successfully acquire large, complex software-intensive systems using a single step (or once-through) strategy, even when the contractor team uses an iterative development life cycle model. Evolutionary acquisition, where the system is acquired in evolutions (or increments), is more suited to the development of large, complex software-intensive systems, as long as the evolutions are feasibly defined from a software perspective. Best practices for defining executable program evolutions are discussed in paragraph 4.1.5 below.

b) Choose Software-Friendly Points in the Life Cycle for Contract Actions

This best practice involves selecting an acquisition strategy where the major contract actions are positioned at low risk points in the acquisition life cycle from a software perspective. The iterative software development life cycle models where the software is implemented in a series of builds or spirals (e.g., the incremental, evolutionary and spiral models) are best suited for large software-intensive systems since they provide the best risk mitigation opportunities. When defining the acquisition strategy, care must be taken to avoid placing major contract actions, such as selection of a single contractor to develop the system, in the middle of the software development builds/spirals.

Depending upon the particular type of iterative software development life cycle model used, the builds or spirals may begin as early as shortly after the System Design Review (SDR). With any iterative life cycle model, the builds or spirals will generally begin no later than shortly after the System Preliminary Design Review (PDR). At any point in time after the start of the builds/spirals, some builds/spirals may be completed, some may be in development (design, implementation, or testing), and some may not have begun. It is strongly recommended that the selection of a single development contractor team be held following SDR, at the beginning of NSS acquisition phase B (Preliminary Design). The software development may be adversely affected if parallel development contracts are extended through Phase B and will certainly be adversely affected if parallel development contracts are extended through NSS acquisition phase C (Complete Design). This is because these acquisition strategies will generally result in postponing the start of the software builds/spirals until after the single contractor team is selected. This delay occurs for two reasons: 1) to avoid paying for wasted software development effort by the losing development contractor team(s), and 2) to minimize the advantage of the parallel development contractor teams already on contract if there are new bidders in a free and open competition for the next phases. Due to the size and complexity of the software in NSS space systems, postponing the start of the builds or spirals can result in an infeasible software development schedule.

The APB is established at KDP B before the program enters NSS acquisition phase B. This baseline needs to be derived from as accurate an estimate as possible of the software size, effort, cost, and schedule at that point in the life cycle. For this reason, it is important for a large, complex software-intensive system to be at an SDR level of maturity before KDP B. According to NSS Acquisition Policy 03-01 (dated 27 December 2004), KDP B is held following the SDR (see Figure 2 above). This makes the NSS acquisition life cycle superior to the DOD life cycle for large, complex software-intensive systems, since DODI 5000.2 places Milestone B after the System Requirements Review (SRR) instead of after the System Functional Review (SFR), which is the DOD equivalent to the SDR. This means that DOD programs must commit to an APB with a less mature definition of the software cost and schedule.

c) Tailor the Acquisition Model for the Software-Intensive System

This best practice involves the consideration of the software development effort when tailoring the acquisition life cycle model for the program's acquisition strategy. This includes the considerations discussed in paragraphs 4.1.4a) and 4.1.4b) above for using a software-friendly acquisition model and for choosing software-friendly points in the acquisition life cycle for contract actions. Figure 4 depicts an example of an NSS acquisition life cycle model for a small quantity procurement (i.e., without production) tailored to be appropriate for a software-intensive space system. This example shows an acquisition strategy calling for major contract actions following KDPs A and B, but not KDP C or thereafter. For simplicity, this figure depicts a single step acquisition strategy or only the first evolution of an evolutionary acquisition strategy. (An example of a corresponding DOD acquisition life cycle model without production is shown in the figure for reference.)

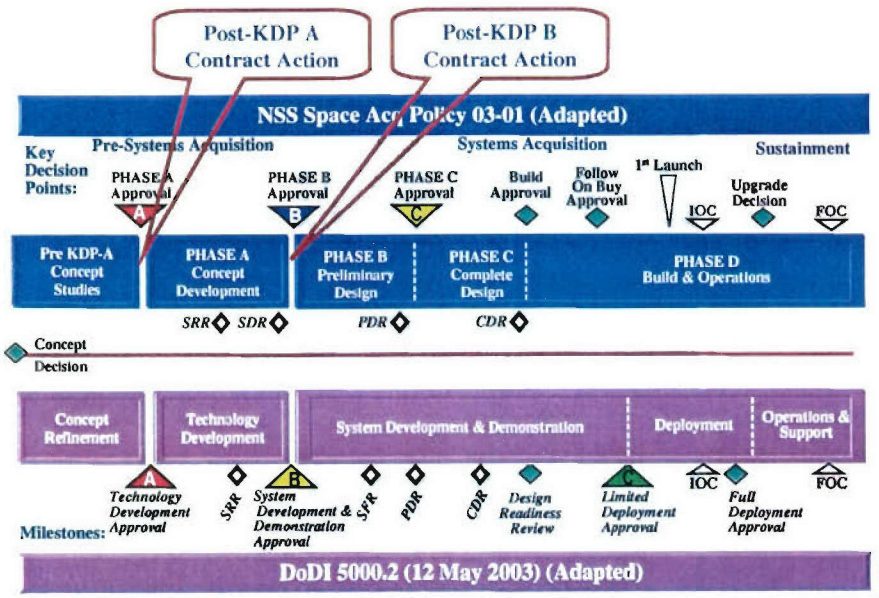

Figure 4. Example NSS Acquisition Life Cycle Model (Tailored for Software-Intensive Systems without Production).

4.1.2 Developing the Initial System Definition

There are two software acquisition best practices for developing the initial system definition: a) ensuring the initial capabilities appropriately include software and b) ensuring the concept of operations appropriately includes software.

a) Ensure the Initial Capabilities Appropriately Include Software

This best practice involves the inclusion of software in the definition of the initial system capabilities. Under the new DOD capabilities generation process, the user organization for the space system is generally assigned responsibility for preparing the Initial Capabilities Document (ICD). However, the acquisition team[1] participates with the users in defining these capabilities. Acquisition team systems engineers provide information to the users from results of Pre-KDP A concept studies related to the feasibility of various functional and performance capabilities, provide input to the users related to the wording of the initial capabilities, and iterate with the users in multiple reviews of the evolving ICD throughout the Pre-KDP A Phase until an approved ICD is achieved.

The appropriate inclusion of software in the definition of the initial system capabilities is essential for reducing downstream software risk. Of particular importance is ensuring that the initial capabilities are feasible from a software perspective and that they are stated in such a way that they cannot be misinterpreted to apply only to hardware. Examples of capabilities that must be both hardware and software inclusive are system performance capabilities (e.g., mission performance timelines, accuracies); system dependability, reliability, maintainability, and availability; system supportability, including testability and integrated system diagnostics; system safety; system security, including information assurance; human systems integration; and interoperability. In addition, the DOD Architecture Framework (DODAF) product views included in the ICD must be both hardware and software inclusive. The most effective application of this best practice requires the inclusion of software systems engineers on the acquisition systems engineering team responsible for working with the user organization on the ICD.

b) Ensure the Concept of Operations Appropriately Includes Software

This best practice involves the inclusion of software in the definition of the system Concept of Operations (CONOPS). For military space systems, the user organization is responsible for preparing the CONOPS document. However, the acquisition team participates in defining the CONOPS by providing operational concept information to the users from the results of Pre-KDP

[1] The government acquisition team for most space systems consists of government acquisition personnel (military and civilian), Federally Funded Research and Development Center (FFRDC) personnel, and personnel from Systems Engineering and Technical Assistance (SETA) contractors. To simplify terminology, the term "acquisition team" will be used throughout this report to refer to members of the government acquisition team, unless the context requires different terminology for clarity.

A concept studies and by iterating with the users in multiple reviews of the evolving system CONOPS until an approved CONOPS document is achieved.

It is extremely important that the system CONOPS be feasible from a software perspective, since for software-intensive space systems, the CONOPS is actually embodied in the software with which the operators must interact. The software's user interface is the foundation for enabling the allocated number of operators to be able to perform their operational tasks within the required timelines under operational workloads. The appropriate inclusion of software in the definition of the system CONOPS is thus essential for reducing downstream software risk. The most effective application of this best practice requires the inclusion of operations-cognizant software systems engineers and software-cognizant human systems integration personnel on the acquisition systems engineering team responsible for working with the user on the definition of the CONOPS.

4.1.3 Developing the Initial Government Architecture Concepts

There are three software acquisition best practices for developing the initial government architecture concepts: a) performing software-inclusive architecture trade studies, b) including software in the evaluation of architecture concepts, and c) selecting a set of integrated hardware/software concepts.

a) Perform Software-Inclusive Architecture Trade Studies

This best practice involves the inclusion of the software architecture along with and as an integral part of the system architecture trade studies. Architectural decisions made in isolation from the software implications of those decisions can adversely affect program executability in terms of the ability to develop the required software within the program's cost and schedule baselines. Decisions that involve software as well as hardware include space-ground and hardware-software allocation trades, onboard computer hardware sizing, and reuse of major legacy system components. Of particular importance for reusing legacy software is performing a cost-benefit trade study that considers the architectural constraints, any re-engineering needed, and the life cycle maintenance implications of the legacy software. The most effective application of this best practice requires the participation of acquisition team personnel knowledgeable in space system software architecting along with the space system hardware-knowledgeable engineers.

Including software architecture as an integral part of the system architecture trade studies supports the establishment of a robust, integrated government hardware/software architecture baseline, which is essential in order to obtain an accurate cost and schedule estimate for the entire system. If the hardware and software architectures are not integrated and consistent with each other, the cost and schedule estimates most likely will be too low since there is a high probability that an inconsistent, non-integrated architecture will not satisfy all of the system requirements. The software portion of the integrated hardware/software architecture baseline must consider Commercial Off-the-Shelf (COTS), reuse and newly developed software for all space and ground elements.

b) Include Software in the Evaluation of Architecture Concepts

This best practice involves the inclusion of software considerations when evaluating proposed system architecture concepts. During the Pre-KDP A phase, multiple architecture concepts are usually considered and evaluated as part of architecture trade studies. When establishing criteria for these evaluations, it is essential that software-related evaluation criteria be included. Some examples of important software-related evaluation criteria are: 1) the amount of space versus ground software to be developed (new, modified reuse, and unmodified reuse) for space-ground trades; 2) the COTS software availability and maturity for potential computer hardware platforms, including development tool suites as well as COTS software to be incorporated into the operational system or used for operations support; 3) the amount of re-engineering necessary for legacy software architectural components; 4) the ability of the architecture to support COTS software refresh and evolution; and 5) the ability of the architecture to support computer resource usage growth (e.g., processor throughput, memory, storage, network and channel bandwidth) due, for example, to increased operational workloads, refined algorithms, additional functionality to satisfy new requirements, or fixes to defects.

Life cycle cost should always be included as an important criterion for architecture evaluation. It is essential that full software costs be included in any life cycle cost analysis. In addition to the cost of developing new software, life cycle cost analysis must include the cost of re-engineering, integrating and verifying legacy software, and the cost of developing "glue" code and integrating COTS software. In addition, life cycle cost analysis must include the cost of COTS software refresh throughout development and sustainment. It is essential that the life cycle cost analysis for candidate system architectures fully reflects the true cost of software; otherwise there is a significant risk of selecting an architecture that later causes severe budget problems during development or sustainment.

c) Select a Set of Integrated Hardware/Software Architecture Concepts

This best practice involves establishing a robust, integrated government hardware/software set of architecture concepts that form the initial government architecture baseline. An integrated set of hardware/software architecture concepts is necessary to provide a firm basis for program cost and schedule estimates (see paragraph 4.1.4 below). The selection of the integrated hardware/software architecture concepts should be based on software-inclusive trade studies and software-inclusive evaluation criteria, as described in paragraphs 4.1.3a) and 4.1.3b) above.

However, when evolutionary acquisition is used, there are additional software-related issues that must be considered when selecting the integrated hardware/software architecture concepts. In this case, the selected architecture concepts must be able to be implemented in a well-defined set of evolutions that provide increasing system capability. This means that the architecture must be able to grow with each successive evolution into its conceived final state with as little rework as possible from one evolution to the next. The ability to easily integrate each successive evolution with the previous evolutions (and with any legacy systems, as applicable) is an especially important characteristic of the software architecture for a system being procured using

evolutionary acquisition. Without this characteristic, the risk of costly and time-consuming downstream rework is very high.

4.1.4 Developing the Initial Government Cost and Schedule Baseline

There are three software acquisition best practices for developing the initial government cost and schedule baseline: a) determining realistic software size estimates for each evolution, b) determining realistic software effort and cost estimates for each evolution, and c) determining realistic software schedule estimates for each evolution.

a) Determine Realistic Software Size Estimates for Each Evolution

This best practice involves making software size estimates that are as realistic as possible, given the early state of the program. Software size is the largest driver of the program's ultimate software cost and schedule. The software size estimates must be based on the government software architecture baseline (see paragraph 4.1.3c) above) and historical software size data from similar past programs. In the Pre-KDP A phase, some actual software size data may be available from prototypes built by concept study contractors[2]. However, for the most part, software size will need to be estimated using historical data from similar functions on other programs, similarity analysis, and engineering judgment. For evolutionary acquisition, the size of the software must be estimated for the program as a whole and separately for each evolution since each evolution must be executable (see paragraph 4.1.5 below).

The software size estimates need to include all software that must be developed for the program. Software size must include the size of all operational onboard software (e.g., spacecraft, payload, communications), all operational ground software (e.g., telemetry, tracking and commanding; mission planning and mission management; mission data processing; automated ground control; orbit determination; and software providing services and infrastructure to other applications), and all software needed for supporting development, manufacturing, testing and operations and for verifying requirements (e.g., simulators, automated test equipment software, test driver software, database management software for onboard constants, and problem report tracking software). Size estimates must be provided for both newly developed code and legacy reuse code. For legacy code, as appropriate to its modified or unmodified status, estimates of the percentage of redesign, recode, and retesting must be made.

For the NSS environment, most historical software size data is available in units of SLOC. However, there are software development tasks that are not easily measured by SLOC, such as developing Graphical User Interface (GUI) screens, building database tables, and integrating COTS software. Appropriate size estimates must be made for these tasks as well, depending upon the particular software cost model to be used.

---

[2] Large, complex space systems usually require a large contractor team consisting of the prime contractor and multiple team members (e.g., subcontractors, other intra-corporate organizations, vendors), even for early acquisition phase contracts. To simplify terminology, the term "contractor" will be used throughout this report to refer to the entire contractor team, unless the context requires different terminology for clarity.

An important part of this best practice is that the government's software size estimates should be independent; that is, the acquisition team should perform its own software size estimation and not rely solely upon contractor estimates. Data from contractors should provide important input into the acquisition team's estimation process. However, the independence of the acquisition team's estimates is necessary to counteract the biases present in contractors' size estimates (e.g., underestimation of amount of new code to be developed, overestimation of amount of COTS and reuse code).

b) Determine Realistic Software Effort and Cost Estimates for Each Evolution

This best practice involves making realistic software effort and cost estimates based upon realistic software size estimates (see paragraph 4.1.4a) above), valid historical data, and appropriate use of software cost models. As with software size, software effort and cost must be estimated separately for each evolution when evolutionary acquisition is used.

Care should be taken to ensure that the government's effort and cost estimates include all work that must be accomplished and all costs that must be incurred in order to develop the software portion of the software-intensive system for the entire development life cycle, from system requirements definition through transition to operations and maintenance. This includes effort for developing new software, re-engineering and modifying reuse software, and integrating the new, reuse, and COTS software into the system. Estimates must also be made of effort for work not easily estimated by software cost models (e.g., database and knowledge base population), as well as for work not easily sized by SLOC (e.g., GUI screen and database table development). The costs of purchasing COTS software licenses, incorporating COTS upgrades, and obtaining COTS vendor support must also be included in the cost estimates.

Similar to software size estimates (as described in paragraph 4.1.4a) above), the government's software effort and cost estimates should be independent; that is, the acquisition team should perform its own software effort and cost estimation and not rely solely upon the concept study contractors' estimates. Effort and cost estimation data from the contractors should provide important input into the acquisition team's estimation process. However, the independence of the acquisition team's estimates is necessary to counteract the biases present in the contractors' effort and cost estimates (e.g., overly optimistic parameters used in the software cost models and overly optimistic productivity estimates). Current Air Force policy requires estimating software cost at the 80% likelihood level or higher (i.e., the probability of successful completion of the software development within the estimated cost is greater than or equal to 80%) [USECAF2].

In addition, the software cost analysis must include analyzing the government funding profile for adequacy of funding for each fiscal year of software development. Funding profiles that are extremely constrained in the early years of software development cause important activities to be postponed and shortcuts to be taken that increase the risk of downstream rework and software cost and schedule overruns.

Finally, realistic estimates of software cost risk must be included in the software cost estimates based on the uncertainty in the parameters input to the estimating process. Software cost risk is

always large in the Pre-KDP A phase, due to the inaccuracy of the software size estimates early in the program.

c) Determine Realistic Software Schedule Estimates for Each Evolution

This best practice involves making realistic software schedule estimates based on realistic software size and effort estimates (see paragraph 4.1.4b) above) and appropriate use of software cost models. As with software size, effort and cost, software schedules must be estimated separately for each evolution when evolutionary acquisition is used.

Software schedule estimates are provided by most software cost models based on the amount of effort required to develop the software. However, when estimating software schedules, all work must be included in the schedule, including the work not easily estimated by software cost models. In addition, realistic estimates of software schedule risk must be included in the software schedule estimates based on the uncertainty in the parameters input to the estimating process. Software schedule risk is always large in the Pre-KDP A phase, due to the inaccuracy of the software size estimates early in the program.

In addition, as with software size, effort and cost, the government's software schedule estimates should be independent; that is, the acquisition team should perform its own software schedule estimation and not rely solely upon the concept study contractors' estimates. Data from the contractors should provide important input into the acquisition team's estimation process. However, the independence of the acquisition team's estimates is necessary to counteract the biases present in the contractors' estimates (e.g., infeasible parallelism and overlap of scheduled activities).

Realistic software cost and schedule estimation not only increases the predictability of the cost and schedule, and, therefore, decreases the probability of overruns; it also contributes to improved quality of the software product. Sufficient time and effort to develop the software products, including performing the quality checks and correcting any identified defects, are essential to reducing downstream rework due to latent software defects. When extreme cost or schedule constraints are imposed upon the development contractor, neither the cost nor the schedule is likely to be met. In addition, the resulting cost or schedule pressure will cause poor software product quality since the contractual cost or schedule constraints will force the developer to take shortcuts in the software development processes. According to the DOD Software Program Manager's Network, "attempts to compress a schedule to less than 80% of its nominal schedule aren't usually successful" [SPMN, p.22]. Developing realistic software cost and schedule estimates in the early acquisition phases will help reduce risk of software cost and schedule overruns and low software quality later in the development effort.

4.1.5 Defining Executable Program Evolutions

There are two software acquisition best practices for defining executable program evolutions: a) considering software implications when defining evolution capabilities and b) considering software implications when defining evolution schedules.

a) Consider Software Implications When Defining Evolution Capabilities

This best practice involves including software considerations when making decisions about the capabilities to be allocated to each evolution when evolutionary acquisition is used. Deciding on evolution capabilities without analyzing the feasibility of the software development required to provide those capabilities can easily lead to the definition of an unexecutable program. For space systems in particular, defining the capabilities strictly in terms of the space segment can lead to a situation where nearly all of the ground software (usually millions of SLOC) must be developed in the first evolution to support the first block of satellites, resulting in severe up-front cost and schedule constraints for the ground software that cannot be met.

To ensure that a feasible evolutionary acquisition strategy is defined, three types of software analyses need to be performed. The first type is an analysis of the software cost and schedule, where the feasibility of developing the required software for each evolution is analyzed to determine whether the evolution's software can be developed within the program's funding profile and schedule constraints for the evolution. This feasibility analysis must include the software cost and schedule estimation risk for each evolution. (See paragraph 4.1.4 above for best practices for estimating software size, effort, cost, and schedule.)

The second type is an analysis of the integrated hardware/software architecture (see paragraph 4.1.3 above) together with the definition of the evolutions' capabilities to determine the feasibility of integrating the software in each subsequent evolution with the software developed in the previous evolution(s). The evolutions' capabilities must be consistent with the ability of the integrated hardware/software architecture to be implemented in the defined evolutionary stages without significant amounts of "breakage" (i.e., large amounts of software that must be discarded or reworked from one evolution to the next). This analysis should also include analyzing the feasibility of integrating each evolution with any legacy system(s), if required.

The third type is an analysis of the impacts of COTS software refresh, legacy software upgrades, and new technology insertion on the defined evolutions' capabilities. On the one hand, each evolution must be capable of supporting COTS software refresh, legacy software upgrades, and new technology insertion without affecting the defined capabilities. On the other hand, the evolutions' capabilities may be dependent upon using new computer system (hardware and/or software) technologies or new features of COTS or legacy software. The likelihood of the readiness of the needed new technologies and the availability of new COTS and legacy software features must be analyzed to determine the feasibility of implementing the evolutions' capabilities on their required schedules.

The most effective application of this best practice requires the inclusion of acquisition team software systems engineers, architects, and cost and schedule analysts on the team responsible for defining the program's evolutionary capabilities.

b) Consider Software Implications When Defining Evolution Schedules

This best practice involves including software considerations in defining the evolution schedules when evolutionary acquisition is used. Deciding on evolution schedules without analyzing the feasibility of the evolutions' software development schedules can easily lead to the definition of an unexecutable program.

Not only must the software cost and schedule of each evolution be feasible (see paragraph 4.1.5a) above), but also cross-evolution interactions must be considered. When evolutions have closely spaced schedules, the software development effort for a particular evolution may overlap with the preceding evolution and/or the succeeding evolution. This creates a high-risk software development effort due to the necessity of developing the software based on moving, and possibly unknown, baselines (i.e., the unfinished software from previous evolution(s)). The expected COTS refresh and legacy software upgrade schedules also need to be considered in defining the evolutions' schedules.

The most effective application of this best practice requires the inclusion of acquisition team software systems engineers and cost and schedule analysts on the team responsible for defining the program's evolutionary capabilities. Including software implications when defining the evolutions' capabilities and schedules is essential for defining an executable evolutionary acquisition strategy for the program.

4.1.6 Developing the Global Acquisition and T&E Strategies

There are four software acquisition best practices for developing the global acquisition and T&E strategies: a) developing plans for computer system technology insertion, b) developing plans for software support, c) developing plans for evaluation of contractor software capability, and d) developing a software-inclusive T&E strategy. The word "global" is used here to mean the acquisition and T&E strategies for all evolutions, when the program is using evolutionary acquisition.

a) Develop Plans for Computer System Technology Insertion

This best practice involves including plans for computer system technology insertion as part of the program's acquisition strategy planning. This includes planning for the refresh of COTS computer hardware and software during the full life cycle, including development and sustainment, and for all evolutions. It also includes evaluating the readiness of new computer hardware and software technologies and planning for their insertion into appropriate evolutions. Ensuring technology readiness at the level needed for use by the program may involve planning to invest program funds for computer system technology maturation.

b) Develop Plans for Software Support

This best practice involves including plans for software support as part of the program's acquisition strategy planning. At the most basic level, this planning involves deciding on the method for obtaining software support (e.g., sole source contract to the developer, competitive sustainment contract, organic maintenance by a logistics organization). However, software adds complexity to planning for support. A strategy must be defined for managing multiple software baselines for operations and development. At times there may be a number of concurrent operational and development baselines to handle the development contractor using iterative life cycle models and the program using the evolutionary acquisition life cycle model, as well as for multiple operational installations. In addition, a strategy must be defined for integrating software maintenance actions made on operational software into the software still under development.

c) Develop Plans for Evaluation of Contractor Software Capability

This best practice involves including plans for performing one or more formal evaluations of contractor software capability as part of the program's acquisition strategy planning. The purpose of a formal evaluation of software capability is to help ensure that a development contractor team is selected that is capable of performing the software development effort required for the program. To support source selection at SMC, there are two methods currently in use for evaluating contractor capability: the Air Force's Software Development Capability Evaluation (SDCE) [AFMC] and the Software Engineering Institute's (SEI's) Standard CMMI® Appraisal Method for Process Improvement (SCAMPI℠)[SEI1][3]. The SCAMPI℠ method is based on the SEI's Capability Maturity Model® Integration (CMMI®) [SEI2 and SEI3][4].

SMC policy requires that each program perform at least one formal evaluation of contractor software capability using one of the above two methods. The evaluation must be performed for the contractor team as a whole, not for each software team member separately, and must evaluate the processes proposed for use by the contractor team on the specific program, not just untailored organizational processes. It is essential that this formal evaluation be performed before the selection of the single development contractor team. Also, the results of the evaluation must be part of the source selection evaluation criteria. For the contractor capability evaluation to be a meaningful discriminator in the source selection, sufficient weight must be given to the evaluation results in the evaluation criteria. Based on the program's acquisition strategy, more than one formal evaluation of contractor software capability may need to be performed (e.g., as part of multiple down-selections between acquisition phases).

d) Develop a Software-Inclusive T&E Strategy

This best practice involves incorporating software when defining the program's T&E strategy. The T&E strategy must cover all T&E concepts for all evolutions and all acquisition phases

[3] SCAMPI is a Service Mark (SM) of Carnegie Mellon University.
[4] CMMI and Capability Maturity Model are registered by Carnegie Mellon University in the U.S. Patent and Trademark Office.

beginning with Phase A, including modeling and simulation, development test and evaluation, and operational test and evaluation.

The program's T&E strategy has important implications for software, since it enables the identification of software required to support the T&E strategy, such as models, simulators, emulators, test driver software, and automated test equipment software. Plans for developing all of the T&E support software must be part of the acquisition strategy, and accurate estimates for this software must be included in the program's software size, effort, cost and schedule estimates (see paragraph 4.1.4 above). In addition, the T&E strategy must include concepts for robust testing of the program's software in all levels of testing, as well as software maturity success criteria.

The most effective application of this best practice requires the inclusion of acquisition team software systems engineers and testers on the team responsible for defining the program's T&E strategy.

4.2 Phase A Space System Software Acquisition Best Practices

The Phase A activities focus on sufficiently maturing the program baseline (technical, cost and schedule) to be ready to enter the formal acquisition process, which occurs after the approval of the APB at KDP B. Figure 5 shows Phase A and its principal objective, which is to develop the information needed to: 1) solidify the program definition to establish an executable program, and 2) update the global acquisition strategy, including the acquisition plans for this and all future evolutions. Large, complex NSS system developments frequently use multiple parallel contracts during Phase A, with the selection of a single development contractor for phases B, C and D.

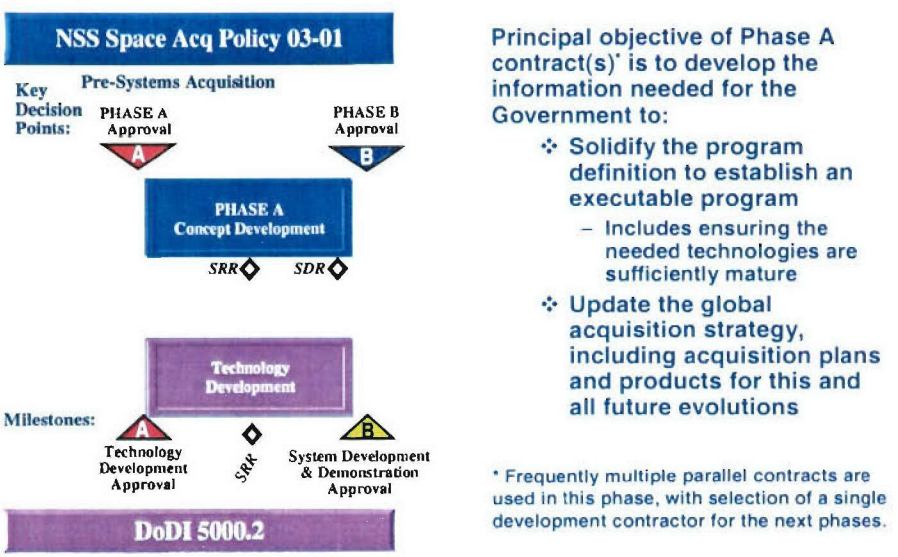

Figure 5. Phase A Principal Objective.

There are six categories of best practices for Phase A as shown in Figure 6. There are three pre-contract award categories of best practices: establishing the requirements baseline (see paragraph 4.2.1), developing the system architectural design (see paragraph 4.2.2), and reducing software risk via the Phase A contracts (see paragraph 4.2.3). There are also three post-contract award categories of best practices: managing the Phase A contracts (see paragraph 4.2.4), refining the system definition (see paragraph 4.2.5), and developing the global acquisition and T&E plans (see paragraph 4.2.6). The software acquisition best practices in each of these categories are discussed below.

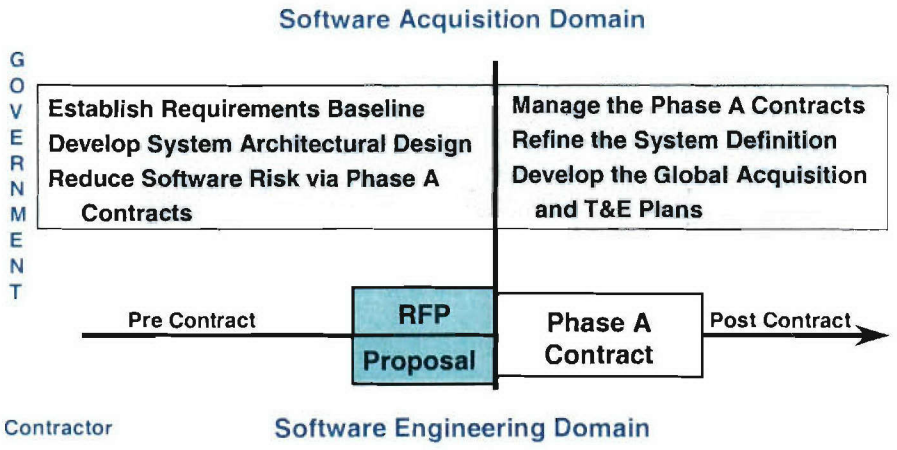

Figure 6. Phase A Software Acquisition Best Practices.

4.2.1 Establishing the Requirements Baseline

There are two software acquisition best practices for establishing the program's requirements baseline: a) including software in the government system performance requirements and b) contracting for delivery of software-inclusive specifications. These best practices are pre-contract award activities for Phase A that are conducted toward the end of the Pre-KDP A phase.

a) Include Software in Government System Performance Requirements

This best practice involves the inclusion of software in the specification of the government's system performance requirements for the system to be developed. The government develops and documents the system performance requirements based on the user organization's ICD and the results of the Pre-KDP A concept studies. At SMC, the document containing the government's system performance requirements is usually called the Technical Requirements Document (TRD), although this name is not universally used. The TRD becomes part of the Phase A contract and is used as the basis for the Phase A contractors' further system requirements analysis and definition.

The appropriate inclusion of software in the definition of the system performance requirements is essential for reducing downstream software risk. Of particular importance is ensuring that the system performance requirements are feasible from a software perspective and that they are stated in such a way that they cannot be misinterpreted to apply only to hardware. Examples of system

performance requirements that must be both hardware and software inclusive are specialty engineering requirements (including dependability, reliability, maintainability, and availability; supportability, including testability and integrated system diagnostics; safety; security, including information assurance; and human systems integration); Key Performance Parameters (KPPs) (e.g., mission performance timelines, accuracies); computer resource reserves (especially onboard margins for processor throughput, memory, storage, and communication bandwidth); other design requirements for evolution and growth; and interoperability (including open systems interface requirements).

Since the system performance requirements become the Phase A contractual requirements, it is very important that they have a complete system perspective, including both hardware and software. The system performance requirements should not be specified so as to reflect only the hardware contribution. For example, including software in the system availability requirements will help ensure that the availability actually experienced during operations will meet the specified requirements. When availability requirements include only hardware, the operationally experienced availability will be significantly less than specified in the requirements due to system failures or inefficiencies caused by software. This can result in the system not being suitable for operations.

The most effective application of this best practice requires the participation of software systems engineers on the acquisition systems engineering team responsible for developing the system performance requirements.

b) Contract for Delivery of Software-Inclusive Requirements Specifications

This best practice involves requiring the Phase A contractors to deliver software-inclusive requirements specifications. This is accomplished by including the System Specification and the Segment Specifications as deliverables in the Phase A Contract's Contract Data Requirements List (CDRL). A standard Data Item Description (DID), the System/Subsystem Specification DID [DOD3], is available for use for these specifications. The System Specification, when finalized and approved by the government, eventually becomes the contractual specification for the development of the system. The System Specification thus eventually contains the contractually compliant technical baseline for the program.

During Phase A, the contractors perform system requirements analysis and definition, refining and elaborating the TRD into a fully elaborated System Specification. The contractors then allocate the system requirements to segments and continue refining and elaborating the allocated requirements into fully elaborated specifications for each segment (e.g., space, ground, user). The results of these requirements analysis, allocation, and elaboration activities are reviewed during Phase A at the System Requirements Review and the System Design Review (see Figure 2). While the DID used for the System and Segment Specifications clearly requires an integrated hardware/software perspective, the government must ensure through proper contract monitoring that the contractors are including both hardware and software in their system and segment requirements analysis and definition processes (see paragraph 4.2.4a) below).

4.2.2 Developing the System Architectural Design

There are two software acquisition best practices for developing the system architectural design: a) contracting for software architecture trade studies and b) contracting for delivery of a system architecture. These best practices are pre-contract award activities for Phase A that are conducted toward the end of the Pre-KDP A phase.

a) Contract for Software Architecture Trade Studies

This best practice involves contracting for software architecture trade studies to be performed along with and as an integral part of the system architecture trade studies as part of the Phase A contract. During the system design activities of Phase A leading up to the System Design Review, the contractors perform detailed system architecture trade studies, make hardware and software architecture decisions, and complete the system design activities for their selected architecture. This best practice ensures that software has been given adequate consideration in these activities in order to reduce the risk of proceeding with an inconsistent, non-integrated architecture that may not satisfy all of the system requirements.

As described in paragraph 4.1.3 above, architectural decisions made in isolation from the software implications of those decisions can adversely affect program executability in terms of the ability to develop the required software within the program's cost and schedule baseline. Decisions that involve software as well as hardware include space-ground and hardware-software allocation trades, onboard computer hardware sizing, and reuse of major legacy system components. Of particular importance is performing a cost-benefit trade study for reusing legacy software that considers architectural constraints of the legacy software, any re-engineering needed, and any life cycle maintenance implications. The software portion of the integrated hardware/software architecture baseline must address COTS, reuse, and newly developed software for all space and ground elements.

b) Contract for Delivery of a System Architecture

This best practice involves requiring the Phase A contractors to deliver the system architecture. This is accomplished by including System/Segment Design Document (SSDD) as a deliverable in the Phase A Contract's CDRL. The SSDD must be required to present an integrated hardware/software architecture, defined by multiple architecture views. The software portion of the system architecture must cover the newly developed, reuse, and COTS software components for all space and ground elements, including all non-deliverable operational support software (e.g., simulators). A standard DID, the System/Subsystem Design Document DID [DOD4], is available for use but must be tailored to add the required DOD or NRO architecture framework views.

A robust SSDD will reduce risk by providing a firm basis for the contractors' proposal for the next acquisition phase and for continuing with the preliminary design. While the tailored DID used for the SSDD will clearly require an integrated hardware/software perspective, the government must ensure through proper contract monitoring that the contractors are appropriately including both hardware and software in their system architecture and design processes (see paragraph 4.2.4b) below).

4.2.3 Reducing Software Development Risk via the Phase A Contracts

There are two software acquisition best practices for reducing software development risk via the Phase A contracts: a) contracting for software product risk reduction and b) contracting for software process risk reduction. These best practices are pre-contract award activities for Phase A that are conducted toward the end of the Pre-KDP A phase.

a) Contract for Software Product Risk Reduction

This best practice involves including specific tasks in the Phase A contract to reduce risk in the eventual development of the software products. This means that the government is explicitly spending funds during Phase A to mitigate downstream risk in developing the software products.

The software product risk reduction tasks are dependent upon the system performance requirements and program acquisition strategy. Examples of these tasks are studies or prototype development of high-risk areas for the software, such as new mission data processing algorithms or new mission planning/mission management concepts. Another important task that should begin during Phase A is simulator development, especially the end-to-end mission simulator, which needs to begin development very early in the life cycle. A third area of tasking is investment in maturing new computer hardware and software technologies to increase their readiness level by the time they are needed for the system. The government must ensure through proper contract monitoring that the contractors are appropriately conducting the contractually required software product risk reduction activities (see paragraph 4.2.4c) below).

b) Contract for Software Process Risk Reduction

This best practice involves including specific requirements in the Phase A contract to reduce risk in the software processes to be used by the contractor team for the entire software development effort, throughout all acquisition phases (A through D). This means that the government is explicitly spending funds during Phase A to mitigate downstream software process risk.

The first Phase A contractual requirement in the area of software processes is to mandate compliance with a robust software development standard. An updated version of MIL-STD-498 [DOD5] for space systems has been developed [ADAMS2]. This updated standard, which is now being used as a compliance document on SMC and NRO programs, contains requirements for all software technical and management processes. During Phase A, this standard should be mandated for two purposes: 1) all software development activities performed during Phase A should be required to be in compliance with the appropriate sections of the standard, and 2) all software processes defined during Phase A for future Phase B, C and D software development activities should be required to be in compliance with the standard.

The second Phase A contractual requirement in the area of software processes is to require delivery of a Software Development Plan (SDP) for the program. This is accomplished by including the SDP as a deliverable in the Phase A contract's CDRL. The SDP must be required to cover all software development activities to be performed throughout Phases A through D for all software being developed for the program (see the types of software described in paragraph 4.1.4a)

above). Furthermore, the SDP must be required to be an integrated plan that addresses the processes to be used by all contractor team members with software development responsibility and how these processes integrate across team member boundaries. A standard Software Development Plan DID [DOD6] is available for use but must be tailored to be consistent with Appendix II of the updated software development standard [ADAMS2].

These Phase A software process risk reduction activities will help the contractor prepare for a software capability evaluation as part of the source selection at the beginning of Phase B when a single development contractor is usually chosen (see paragraph 4.1.6c) above). The government must ensure through proper contract monitoring that the contractors are appropriately conducting the contractually required software process risk reduction activities (see paragraph 4.2.4c) below).

4.2.4 Managing the Phase A Contracts

There are three software acquisition best practices for managing the Phase A contracts: a) ensuring the contractor(s) define software-inclusive requirements specifications, b) ensuring the contractor(s) define an integrated hardware/software architecture, and c) participating with the contractor(s) in software risk reduction. These best practices are post-contract award activities that are conducted throughout the duration of the Phase A contracts.

a) Ensure Contractor(s) Define Software-Inclusive Requirements Specifications

This best practice involves participating with the Phase A contractors in their system requirements analysis and definition activities to ensure that software is being appropriated included (see paragraph 4.2.1 above) in the System and Segment Specifications. The most effective application of this best practice is to include software systems engineers on the acquisition systems engineering team responsible for ensuring that the contractors are using high quality system requirements processes and that the contractor's System and Segment Specifications are correct and complete.

b) Ensure Contractor(s) Define an Integrated Hardware/Software Architecture

This best practice involves participating with the Phase A contractors in their system architectural design activities to ensure that they are developing an integrated hardware/software architecture (see paragraph 4.2.2 above). The most effective application of this best practice is to include software systems engineers and architects on the acquisition systems engineering team responsible for ensuring that the contractors are using high quality system architectural design processes and that the contractor's System/Segment Design Document describes a high quality system design capable of meeting all system performance requirements.

c) Participate with Contractor(s) in Software Risk Reduction

This best practice involves participating with the Phase A contractors in their accomplishment of the software product and software process risk reduction tasks (see paragraph 4.2.3 above). Acquisition team personnel with technical expertise in relevant areas of software product

engineering and algorithms must monitor the contractor's performance on the software product risk reduction tasks. Acquisition team personnel with technical expertise in software process engineering and software project management must participate with the contractor in their definition of the software processes and development of the SDP to ensure that high quality software processes and plans are being prepared for the entire software development effort.

4.2.5 Refining the System Definition

There are two software acquisition best practices for refining the system definition: a) ensuring the required capabilities appropriately include software and b) ensuring the updated CONOPS appropriately includes software. These best practices are post-contract award activities conducted during Phase A.

a) Ensure the Required Capabilities Appropriately Include Software

This best practice involves the inclusion of software in the definition of the required system capabilities. Under the new DOD capabilities generation process, the user organization for the space system is generally assigned responsibility for preparing the Capability Development Document (CDD). However, the acquisition team participates with the users as they elaborate the initial system capabilities documented in the ICD (see paragraph 4.1.2a) above) to the detailed system capabilities documented in the CDD. Acquisition team systems engineers provide information to the users based on results from the Phase A contracts as to the feasibility of various functional and performance capabilities, provide input to the users as to the wording of the capabilities, and iterate with the users in multiple reviews of the evolving CDD throughout Phase A until an approved CDD is achieved.

The appropriate inclusion of software in the definition of the system capabilities is essential for reducing downstream software risk. Of particular importance is ensuring that the system capabilities are feasible from a software perspective and that they are stated in such a way that they cannot be misinterpreted to apply only to hardware. Examples of capabilities that must be both hardware and software inclusive are KPPs (e.g., mission performance timelines, accuracies); system dependability, reliability, maintainability, and availability; system supportability, including testability and integrated system diagnostics; system safety; system security, including information assurance; human systems integration; and interoperability. In addition, DODAF views included in the CDD must be both hardware and software inclusive. The most effective application of this best practice requires the inclusion of software systems engineers on the acquisition systems engineering team responsible for working with the user organization on the CDD.

b) Ensure the Updated CONOPS Appropriately Includes Software

This best practice involves the inclusion of software in any updates to the system CONOPS made during Phase A. For military space systems, the user organization is responsible for any updates made to the approved CONOPS document (see paragraph 4.1.2b) above). However, the acquisition team participates in this process by providing operational concept information to the

users from the results of Phase A contracts and by iterating with the users in reviews of any updates to the system CONOPS until an approved updated CONOPS document is achieved.

It is extremely important that the updated system CONOPS continues to be feasible from a software perspective, since for software-intensive space systems, the CONOPS is actually embodied in the software with which the operators must interact. The software's user interface is the foundation for enabling the allocated number of operators to be able to perform their operational tasks within the required timelines under operational workloads. The appropriate inclusion of software in any updates to the system CONOPS is thus essential for reducing downstream software risk. The most effective application of this best practice requires the inclusion of operations-cognizant software systems engineers and software-cognizant human systems integration personnel on the acquisition systems engineering team responsible for working with the user on any updates to the CONOPS.

4.2.6 Developing the Global Acquisition and T&E Plans

There are four software acquisition best practices for developing the global acquisition and T&E plans: a) updating the software-inclusive program baseline, b) updating the definition of software-friendly evolutions, c) updating the software-specific plans, and d) developing software-inclusive T&E plans. These best practices are post-contract award activities conducted during Phase A in preparation for KDP B. The word "global" is used here to mean the acquisition and T&E plans for all evolutions, when the program is using evolutionary acquisition.

a) Update Software-Inclusive Program Baseline

This best practice involves updating the software-inclusive program baseline (technical, cost, and schedule) based on the additional knowledge gained during Phase A. Using the Phase A TRD, the contractors' system requirements prepared during Phase A, and the user's CDD, the government develops their system performance requirements for the development contract (covering Phases B through D). It is very important that the government's updated requirements baseline continue to include software (see paragraph 4.2.1 above).

The government also needs an updated system architecture baseline in order to have a firm basis for updated program cost and schedule estimates for KDP B. The government develops this updated architecture baseline from its initial architecture baseline and from the Phase A contractors' system designs. The government architecture baseline must be capable of meeting the government's updated requirements baseline. It is very important that the government's updated architecture baseline be an integrated hardware/software architecture that includes all of the newly developed, reuse, and COTS software necessary for the system (see paragraphs 4.1.3 and 4.2.2).

Using the Phase A contractors' software size, effort, cost, and schedule estimates based on their system architectures and the government system architecture baseline, the government must update its software size, effort, cost, and schedule estimates to develop program cost and schedule estimates for KDP B. It is very important that the government's software size, effort, cost, and

schedule baselines continue to be based on realistic estimates with well-founded rationale and proper use of software cost and schedule models (see paragraph 4.1.4 above).

b) Update Definition of Software-Friendly Evolutions

This best practice involves updating the definition of the evolutions, if evolutionary acquisition is being used, based on the additional knowledge gained during Phase A. This includes updating the capabilities to be implemented in each evolution, the schedule for each evolution, and the strategy for integrating the evolutions with preceding evolutions and legacy systems, if applicable. It is very important that the evolutions' defined capabilities and schedules continue to be feasible from a software perspective (see paragraph 4.1.5 above). In addition, the evolution definitions must consider the availability of COTS and legacy software upgrades to be integrated into each evolution and the availability of needed new computer hardware and software technologies.

c) Update Software-Specific Plans

This best practice involves updating the other important software plans that are part of the program's acquisition strategy based on the additional knowledge gained during Phase A. This includes plans for computer system (hardware and software) technology transition, plans for software support, and plans for evaluations of contractor team software capability (see paragraphs 4.1.6a), 4.1.6b) and 4.1.6c) above).

d) Develop Software-Inclusive T&E Plans

This best practice involves developing software-inclusive test and evaluation plans and documenting them in the Test and Evaluation Master Plan (TEMP), based on additional knowledge gained during Phase A. During Phase A, the government must update the T&E Strategy developed during Pre-KDP A (see paragraph 4.1.6d) above), perform the planning necessary to implement that strategy, and prepare the TEMP. The TEMP must cover the planning for all acquisition phases beginning with Phase A, including modeling and simulation, development test and evaluation, and operational test and evaluation, all of which must properly include software. The TEMP also specifies the Measures of Effectiveness and Suitability and Critical Technical Parameters, which must apply to the integrated hardware/software system (i.e., which must not be specified for hardware only).

The program's TEMP has important implications for software since it identifies the software required to implement the T&E plan, such as models, simulators, emulators, test driver software, and automated test equipment software. Plans for developing all of the T&E support software must be part of the acquisition strategy, and accurate estimates for this software must be included in the program's updated software size, effort, cost, and schedule estimates (see 4.2.6a) above). In addition, the TEMP must include plans for robust testing of the program's software in all levels of development and operational testing.

4.3 Software Acquisition Best Practices That Span the Acquisition Life Cycle

There are two software acquisition best practices that span the full acquisition life cycle: a) software acquisition risk management and b) software systems acquisition.

a) Software Acquisition Risk Management

This best practice involves the government performing software acquisition risk management as an integral part of its program acquisition risk management. Software acquisition risk management is an effective mechanism for reducing the impact of potential problems on the acquisition of a software-intensive system. Software acquisition risk management involves a continuous process of risk identification, assessment, prioritization, mitigation, and control throughout the life cycle of the program, from the identification of needed capability through retirement. For evolutionary acquisition, it is important that software acquisition risk management addresses risks both within each ongoing evolution and across the evolutions, since decisions made during one evolution can increase or decrease software acquisition risk in subsequent evolutions.

The use of software acquisition risk management enables the acquisition team to understand its risks (i.e., future problems that might occur) and, where possible, to mitigate the risks that are assessed to have the largest potential impact on the software-intensive system acquisition. Software acquisition risk management is especially important during the early acquisition phases when important program decisions are being made. Risk mitigation efforts put into place early in the acquisition life cycle can be very effective at reducing downstream software risk.

Software acquisition risk management by the acquisition team is necessary in addition to software development risk management by the contractor. The contractor's software development risks are almost always software acquisition risks. However, the government acquisition organization responsible for acquiring the software-intensive system generally has additional risks as well (e.g., risks related to staffing of the acquisition team and risks related to program KDPs). Furthermore, the acquisition team frequently will identify additional software development risks and will assess the importance of the contractor-identified software development risks differently than the contractor.

To be effective, the software acquisition risk management process must be practiced by all acquisition team personnel and at all levels of the software acquisition organization. Large programs frequently have a risk management process at the program level with the top program risks being closely monitored by government program management. Just as frequently, however, the only risk management process practiced on large, complex software-intensive programs is the program level process. An effective software acquisition risk management process must be practiced at every level of the software acquisition organization, including the lowest level of the acquisition team. The acquisition team should use the software acquisition risk management process to identify its software acquisition risks, mitigate and control those risks that are within its scope of control, and elevate those risks with sufficiently high impact to the program level risk management process.

b) Software-Inclusive Systems Acquisition

This best practice involves the inclusion of software acquisition as an integral part of the acquisition of software-intensive space systems. Software acquisition team personnel must be knowledgeable and must participate in the systems acquisition processes throughout the entire life cycle, from the identification of needed capability through retirement. In addition, the software acquisition processes must be consistent and integrated with the systems acquisition processes. It is very important for software acquisition to be an integral part of the system acquisition's early acquisition life cycle activities, as described in paragraphs 4.1 and 4.2 above.

The effective incorporation of software acquisition as an integral part of the systems acquisition processes requires positive action by the government program management. Government program management must establish an environment where the software acquisition is a highly respected part of the program, equivalent in importance to the hardware acquisition. In addition, the government program must be structured to provide effective lines of communication, responsibility, and authority among the software acquisition team and the other program teams involved in the systems acquisition processes.

5. Conclusion

This paper describes a set of software acquisition best practices for the early acquisition life cycle that the authors have identified, through experience, as being significant contributors to the successful acquisition of software-intensive space systems. This paper describes six categories of software acquisition best practices for the earliest acquisition life cycle phase, Pre-KDP A, as follows:

- Defining the Program Life Cycle
- Developing the Initial System Definition
- Developing the Initial Government Architecture Concepts
- Developing the Initial Government Cost and Schedule Baseline
- Defining Executable Program Evolutions
- Developing the Initial Acquisition and T&E Strategies

This paper also describes six categories of software acquisition best practices for acquisition Phase A, as follows:

- Establishing the Requirements Baseline
- Developing the System Architectural Design
- Reducing Software Development Risk via Phase A Contracts
- Managing the Phase A Contracts
- Refining the System Definition
- Developing the Updated Acquisition Strategy and T&E Plan

The software acquisition best practices described in this paper, however, are not a panacea; that is, they do not guarantee a successful acquisition, since there are a great many influences that can adversely affect large, complex software-intensive space system acquisitions. Using best practices, however, can reduce risk in software-intensive system acquisitions. In particular,

software acquisition best practices applied during the early acquisition phases can be very effective at reducing downstream software acquisition and development risk.

References

[ADAMS1] Adams, R. J., S. Eslinger, K. L. Owens, and M. A. Rich, *Software Acquisition Best Practices: Experiences from the Space Systems Domain*, The Aerospace Corporation, Technical Report TR-2004(8550)-1, 30 September 2004.

[ADAMS2] Adams, R. J., S. Eslinger, P. Hantos, Lt. Col. G. A. Newberry, K. L. Owens, L. T. Stephenson, J. M. Tagami, R. Weiskopf, and M. Zambrana, *Software Development Standard for Space Systems*, The Aerospace Corporation, Technical Operating Report TOR-2004(3909)-3537 Revision A, 11 March 2005.

[AFMC] Air Force Materiel Command, *Software Development Capability Evaluation*, Volumes 1 and 2, AFMCP 63-103, 15 June 1994.

[DOD1] Department of Defense, *Operation of the Defense Acquisition System*, Department of Defense Instruction (DODI) 5000.2, 12 May 2003.

[DOD2] Department of Defense, *The Defense Acquisition System*, Department of Defense Directive (DODD) 5000.1, 12 May 2003.

[DOD3] Department of Defense, *System/Subsystem Specification (SSS) Data Item Description*, DI-IPSC-81431A, 10 January 2000.

[DOD4] Department of Defense, *System/Subsystem Design Document (SSDD) Data Item Description*, DI-IPSC-81432A, 10 August 1999.

[DOD5] Department of Defense, *Software Development and Documentation*, Military Standard MIL-STD-498, 5 December 1994.

[DOD6] Department of Defense, *Software Development Plan (SDP) Data Item Description*, DI-IPSC-81427A, 10 January 2000.

[PAULK] Paulk, M., C. Weber, B. Curtis, and M. Chrissis, *The Capability Maturity Model®: Guidelines for Improving the Software Process*, Addison-Wesley, 1994.

[SEI1] Software Engineering Institute. *Standard CMMI® Appraisal Method for Process Improvement (SCAMPI℠), Version 1.1: Method Definition Document*, CMU/SEI-2001-HB-001, December 2001.

[SEI2] Software Engineering Institute. *Capability Maturity Model® Integration℠, Version 1.1, CMMI® for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI®-SE/SW/IPPD/SS, V1.1), Continuous Representation*, (SEI-2002-TR-011), March 2002.

[SEI3] Software Engineering Institute. *Capability Maturity Model® Integration℠, Version 1.1, CMMI® for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI®-SE/SW/IPPD/SS, V1.1), Staged Representation*, (SEI-2002-TR-012), March 2002.

[SPMN] DOD Software Program Manager's Network, *The Program Manager's Guide to Software Acquisition Best Practices*, Version 2.2, June 1998.

[USECAF1] Undersecretary of the Air Force, *National Security Space Acquisition Policy*, Number 03-01, 27 December 2004.

[USECAF2] Undersecretary of the Air Force, *Revitalizing the Software Aspects of Systems Engineering*, Memorandum No. 04A-003, 20 September 2004.

Acronyms and Abbreviations

®	Registered Trademark
Acq	Acquisition
AF	Air Force
AFMC	Air Force Materiel Command
AFMCP	Air Force Materiel Command Pamphlet
APB	Acquisition Program Baseline
CDD	Capability Development Document
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CMMI®	Capability Maturity Model® Integration℠
CMU	Carnegie Mellon University
CONOPS	Concept of Operations
COTS	Commercial Off-the-Shelf
DID	Data Item Description
DOD	Department of Defense
DOD	Department of Defense
DODAF	DOD Architecture Framework
DODD	DOD Directive
DODI	DOD Instruction
Eng	Engineering
FFRDC	Federally Funded Research and Development Center
FOC	Final Operational Capability
GUI	Graphical User Interface
H/W	Hardware
HB	Handbook
ICD	Initial Capability Document
IOC	Initial Operational Capability
IPPD	Integrated Product and Process Development
KDP	Key Decision Point
KPP	Key Performance Parameter
MIL	Military
MOIE	Mission-Oriented Investigation and Experimentation
NASA	National Aeronautics and Space Administration
NRO	National Reconnaissance Office
NSS	National Security Space
PDR	Preliminary Design Review
RFP	Request for Proposal
S/W	Software
SCAMPI℠	Standard CMMI® Appraisal Method for Process Improvement
SDCE	Software Development Capability Evaluation
SDP	Software Development Plan

SDR	System Design Review
SE	Systems Engineering
SEI	Software Engineering Institute
SETA	Systems Engineering and Technical Assistance
SFR	System Functional Review
SLOC	Source Lines of Code
SM	Service Mark
SMC	Space and Missile Systems Center
SPMN	Software Program Manager's Network
SRR	System Requirements Review
SS	Supplier Sourcing
SSDD	System/Segment Design Document
STD	Standard
SW	Software Engineering
T&E	Test and Evaluation
TEMP	Test and Evaluation Master Plan
TOR	Technical Operating Report
TR	Technical Report
TRD	Technical Requirements Document
U.S.	United States
USAF	United States Air Force
USECAF	Undersecretary of the Air Force

THE AEROSPACE CORPORATION
2350 E. El Segundo Boulevard
El Segundo, California 90245-4691
U.S.A.